Smoothed Particle Hydrodynamics in one spatial dimension

- we aim at writing a 1D SPH code that solves the following equations of hydrodynamics for a set of N particles

$$\frac{dx_i}{dt} = v_i$$

$$\frac{dv_i}{dt} = -\sum_j m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right) \frac{dW_{ij}}{dx_i}$$

$$\frac{d\rho_i}{dt} = \sum_j m_j \left(v_i - v_j \right) \frac{dW_{ij}}{dx_i}$$

$$\frac{de_i}{dt} = \frac{1}{2} \sum_j m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right) \left(v_i - v_j \right) \frac{dW_{ij}}{dx_i}$$

$$P = \left(\gamma - 1 \right) \rho e$$

$$W_{ij} = W\left(\left|x_i - x_j\right|, h_i\right)$$

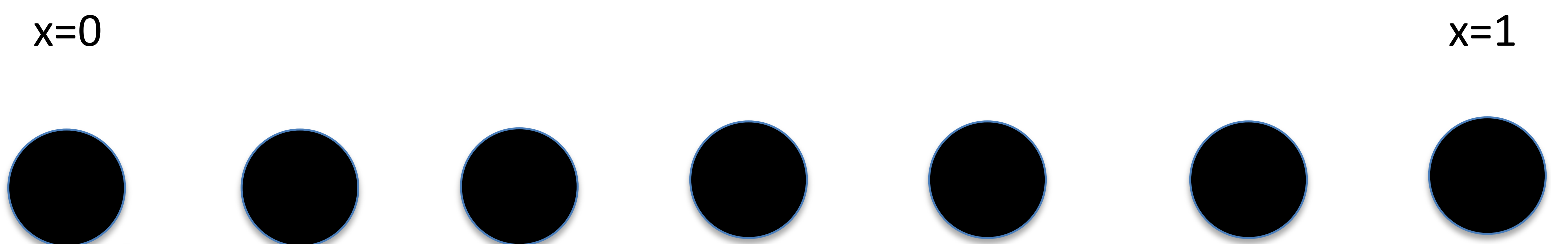
$$\frac{dW_{ij}}{dx_i} = \frac{dW(r, h_i)}{dr} \frac{dr}{dx_i}$$

$$r = \left| x_i - x_j \right|$$

$$W(r,h) = \frac{2}{3h} \begin{cases} 1 - \frac{3}{2}\left(\frac{r}{h}\right)^2 + \frac{3}{4}\left(\frac{r}{h}\right)^3 & 0 \le \frac{r}{h} \le 1 \\ \frac{1}{4}\left(2 - \left(\frac{r}{h}\right)\right)^3 & 1 \le \frac{r}{h} \le 2 \\ 0 & otherwise \end{cases}$$

Smoothed Particle Hydrodynamics in one spatial dimension: blastwave

- we want to simulate a supernova blast wave:
  - the initial conditions are given by placing those N particles down on the interval [0,1] using a spacing dx=1/(N-1)
  - set the thermal energies of all particles to e=1e-5, only the central particle gets e=1 hence triggering a blast wave
  - use an adiabatic coefficient of g=7/5 for this test



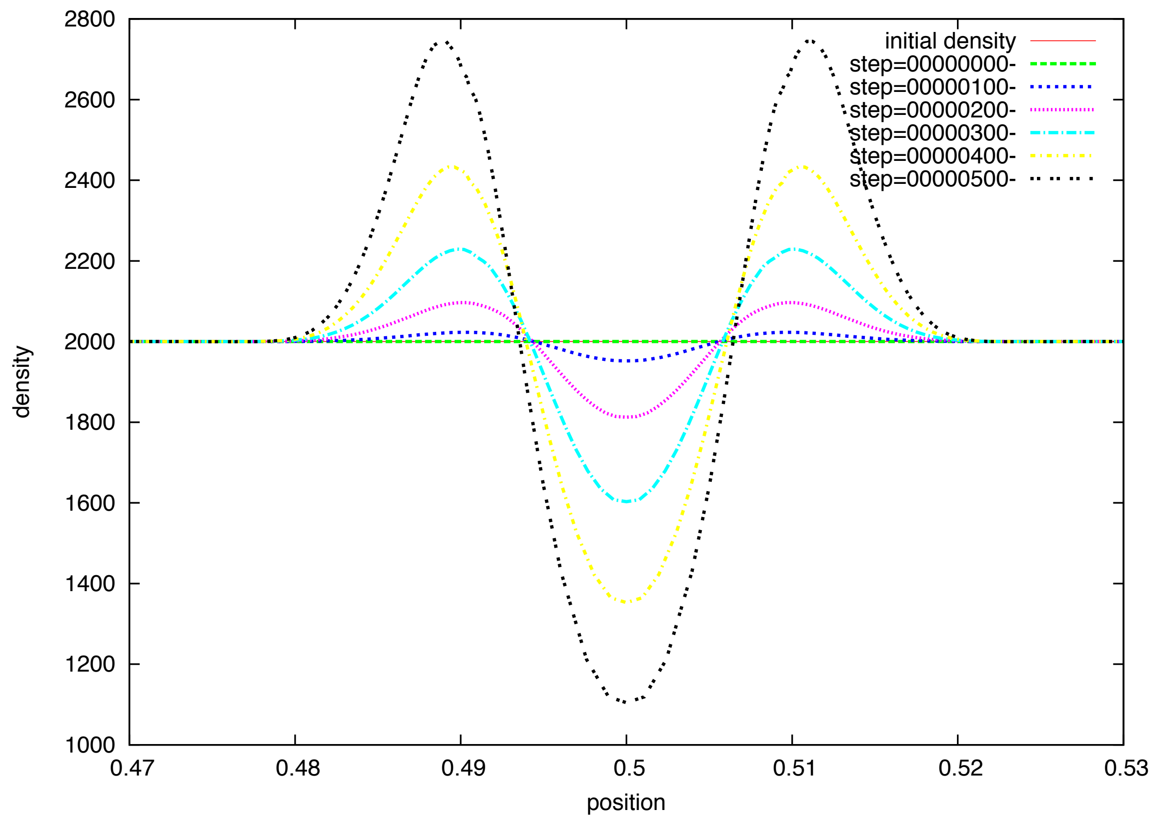
m	1
x	[0,1]
v	0
rho	m/dx
e	1e-5

only for the central particle set e=1

Smoothed Particle Hydrodynamics in one spatial dimension: blastwave

- for the following setup
  - `NPART = 2000` (number of gas particles)
  - `NSTEPS = 500` (number of integration steps)
  - `NSPH = 25` (number of SPH neighbour particles used with the solver)
  - `Tend = 0.05` (end-time of the integration, Tstart=0)

you should be able to generate a plot similar to this one.



coding tips

1.loop (parallel): calculate the coefficients for each particle *i*

$$v\_coeff_i = -\sum_j m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right) \frac{dW_{ij}}{dx_i}$$

$$rho\_coeff_i = \sum_j m_j \left(v_i - v_j \right) \frac{dW_{ij}}{dx_i}$$

$$e\_coeff_i = \frac{1}{2} \sum_j m_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2} \right) \left(v_i - v_j \right) \frac{dW_{ij}}{dx_i}$$

Note: W_{ij} (and hence dW_{ij}/dx_i) is zero for $|x_i - x_j| > 2h_i$

coding tips

2.loop (parallel): use coefficients to perform integration step for each particle *i*

$$x_i \mapsto x_i + v_i \Delta t$$
$$v_i \mapsto v_i + v\_coeff_i \Delta t$$
$$\rho_i \mapsto \rho_i + \rho\_coeff_i \Delta t$$
$$e_i \mapsto e_i + e\_coeff_i \Delta t$$

coding tips

each particle is a structure containing the actual quantities as well as the coefficients

```
struct particle {

double x;
double v;
double rho;
double e;
double h;

double v_coeff;
double rho_coeff;
double e_coeff;

};
```

coding tips

some pseudo C-code for main routine

```
main()
{
	initial_conditions();

	while-loop for the time stepping()
		{
			set_h();
			getSPHcoefficient();
			performIntegration();
			updateTimeCounter();

			WriteOutputFile();

		}
}
```

coding tips

some pseudo C-code for main routine

```
main()
{
    initial_conditions();

    while-loop for the time stepping()
        {
            set_h();  ──────→  how to set the individual h-values?
            getSPHcoefficient();
            performIntegration();
            updateTimeCounter();

            WriteOutputFile();

        }
}
```

coding tips

how to find the NSPH nearest neighbours of particle *i* ????

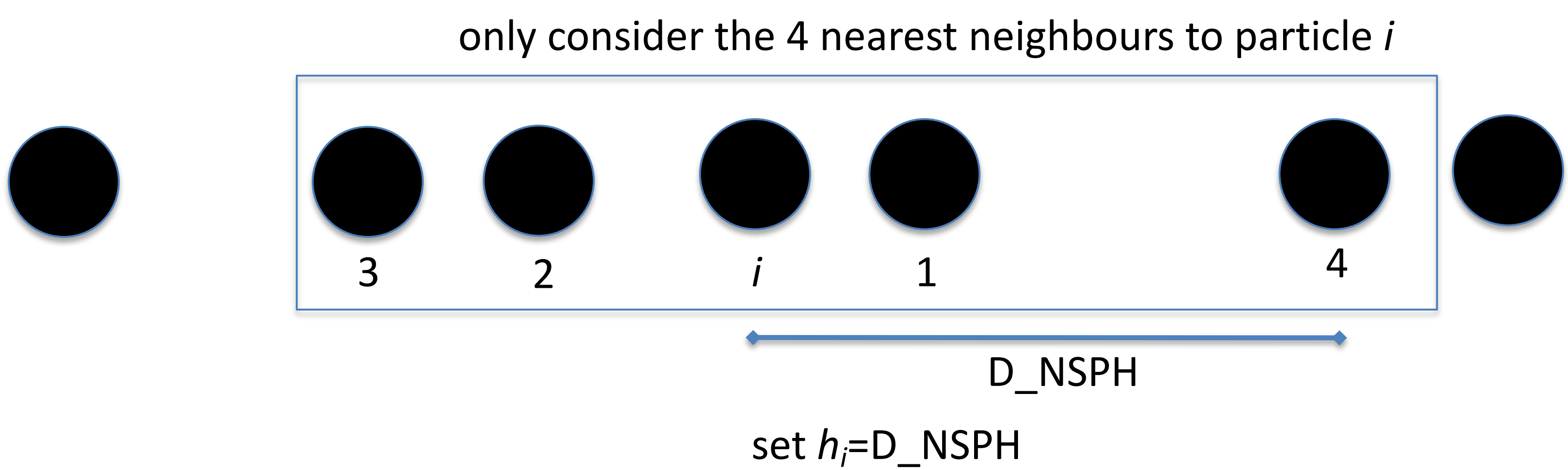
1. calculate the distance of every particle *j* to particle *i*
2. sort the particles with respect to distance
3. only consider the first NSPH particles in this ordered list
4. set *h* to be the distance to particle NPSH

coding tips

how to find the NSPH nearest neighbours of particle *i* ????

1. calculate the distance of every particle *j* to particle *i*
2. sort the particles with respect to distance
3. only consider the first NSPH particles in this ordered list
4. set *h* to be the distance to particle NPSH

example: NSPH=4

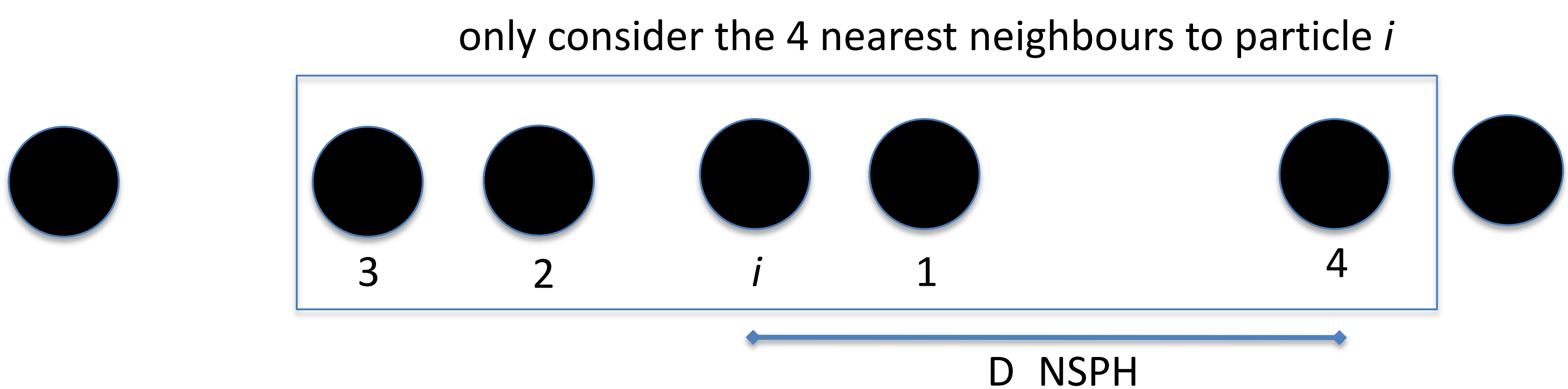


coding tips

how to find the NSPH nearest neighbours of particle *i* ????

1. calculate the distance of every particle *j* to particle *i*
2. **sort the particles with respect to distance?????**
3. only consider the first NSPH particles in this ordered list
4. set *h* to be the distance to particle NPSH

example: NSPH=4



set h_i=D_NSPH

coding tips: utility.c

utility.c

- contains indexx()
- contains SIGN(x) giving you the signum of x (required when calculating dr/dx_i!)
- contains shortcuts pow2(x) for ((x)*(x))
- etc.

coding tips: indexx()

```
indexx(long N, double *array, long *index)
```

input:

N = length of your array

array = pointer to the array that you want to have sorted (minus 1)

output:

index = a pointer to a long integer array (minus 1)

array	index
15	2
0.3	4
10	7
5	3
37.4	6
12	1
8	5
98	8

<u>how to use index():</u>

array[index[0]-1] < array[index[1]-1] < array[index[2]-1] < ...

(all these 'minus 1' will be explained on the next page...)

coding tips: indexx()

```
indexx(long N, double *array, long *index)
```

peculiarity for C implementation from Numerical Recipes:
arrays are addressed in range from [1,N]!

one therefore needs to strangely adjust the usage of the routine as follows:

```
...
array = (double *) calloc(N, sizeof(double));
index = (long *) calloc(N, sizeof(long));

// fill array from array[0] to array[N-1] with distances of all particles to particle i
...

// call indexx() to obtain the index() array
indexx(N, array-1, index-1);  // trick: pass 'pointer - 1'

// when using index[] you must also subtract '-1' to obtain index range [0,N-1]

// this loops prints the array[] in a correctly ordered way now
for(i=0; i<N; i++) {
        fprintf(stderr,"%ld %lf\n", i, array[index[i]-1] );
}
```

possible improvements

- symmetrize kernel: $W_{ij} \rightarrow \frac{1}{2}\left(W(\left|x_i - x_j\right|, h_i) + W(\left|x_i - x_j\right|, h_j)\right)$
- smooth initial explosion
- periodic boundary conditions
- 2^{nd} order Runge-Kutta integration